09

Deployment

Transcrição

Criamos o arquivo de configuração YAML anteriormente para termos o objeto `Pod` com o container da aplicação web no *cluster*, gerenciado pelo Kubernetes. Esperávamos que, caso algo acontecesse com ele, o Kubernetes percebesse que o estado atual do *cluster* foi alterado, e então, buscasse alguma forma de recriar o objeto, deixando o *cluster* igual ao estado previamente configurado em `aplicacao.yaml` .

Mas não foi isso que aconteceu quando deletamos o `Pod` , causando a remoção permanente do nosso *cluster*. Isto ocorre pois os `Pod` são os **objetos mais básicos existentes no Kubernetes**.

Então, quando trabalhamos com eles, não criamos um `Pod` diretamente. É preciso **abstrair este objeto do Kubernetes em outro, capaz de oferecer mais recursos**. Um deles é justamente o de informar ao Kubernetes que queremos que a aplicação tenha sempre um `Pod` rodando com o container junto à aplicação web.

O objeto que utilizaremos para a abstração do `Pod` receberá o nome de ***Deployment***. Ele nos auxiliará a obter camadas adicionais, garantindo a existência constante de um `Pod` com container web, como queremos.

Como não existe um objeto `Container` no Kubernetes, utilizamos o menor objeto existente (o `Pod`), que além disto é o mais básico, portanto não adiciona a camada de estado desejado da nossa aplicação para o Kubernetes gerenciar.

Ou seja, é necessário colocar o objeto `Pod` em outro, o `Deployment` , o qual possui mais recursos e conseguirá adicionar este estado desejado da nossa aplicação. A partir de então, o Kubernetes saberá que sempre queremos ter um `pod` rodando em nosso *cluster*.

Para criarmos este novo objeto, é necessário um novo arquivo de configuração YAML. Vamos voltar ao terminal e digitar o seguinte:

```
atom deployment.yaml
```

Isto faz com que se abra uma janela do Atom com o novo arquivo. A estrutura dos objetos criados no Kubernetes são bem parecidas umas das outras. Assim, teremos:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aplicacao-deployment
  labels:
    app: aplicacao
spec:
  replicas: 1
  selector:
    matchLabels:
      app: aplicacao
  template:
    metadata:
      labels:
        app: aplicacao
```

```
    spec:
      containers:
      - name: container-aplicacao-loja
        image: rafanercessian/aplicacao-loja:v1
        ports:
        - containerPort: 80
```

Sendo que o padrão no objeto `Deployment` será aquele similar ao tipo `Pod`, de onde copiaremos e colaremos as informações. Isto é, toda a informação que vem após `template` no código acima veio de `aplicacao.yaml`, do objeto `Pod` que está abstraindo o container web.

Pelo fato de estarmos realizando a abstração do objeto `Pod` em `Deployment`, é preciso um ajuste na parte do nome que havíamos dado ao `Pod`, por isto o alteraremos e o colocaremos junto a uma **etiqueta de identificação**. Teremos o seguinte:

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: aplicacao-deployment
spec:
  template:
    metadata:
      labels:
       name: aplicacao-pod
    spec:
      containers:
        - name: container-aplicacao-loja
          image: rafanercessian/aplicacao-loja:v1
          ports:
            - containerPort: 80
```

Feito isso, salvaremos o arquivo de configuração YAML, e queremos que o objeto `Deployment` seja implementado em nosso *cluster*, que está abstraindo `Pod`, que por sua vez abstrai o container da aplicação web.

Para nos comunicarmos com o *cluster*, utilizaremos o seguinte comando:

```
kubectl create -f deployment.yaml
```

Com "Enter" teremos a informação de que "aplicacao-deployment" foi criado! Vamos perguntar ao *cluster* se ele de fato possui algum `pod` em execução?

```
kubectl get pods
```

Com o comando acima, confirmamos a existência do `Pod` sendo abstraído pelo objeto `Deployment` e executado no *cluster*. Esperamos que, por estarmos utilizando-o, o Kubernetes saiba que precisará gerenciar esta aplicação.

Caso ocorra algum problema que impeça o funcionamento do `Pod`, esperamos que o Kubernetes tome alguma providência para consertá-lo. Repetiremos, então, o procedimento de remoção do mesmo para verificar o que acontece:

```
kubectl delete pods aplicacao-deployment-17598552-2g9xp
```

Deste modo obteremos a informação de que o `pod` foi deletado com sucesso. Havíamos feito isto na etapa anterior, o que resultou na remoção permanente do objeto. Vamos ver se desta vez o Kubernetes percebeu o estado atual do *cluster*, já que acabamos de configurar o arquivo YAML para que sempre haja um container com a aplicação web rodando.

Para tal, digitaremos:

```
kubectl get pods
```

Isto demonstra que garantimos o gerenciamento por parte do Kubernetes, que criou um novo `pod` denominado `aplicacao-deployment-17598552-74tm4` para substituir aquele que foi removido!

Agora queremos acessar a nossa aplicação. Vamos trabalhar nisso na sequência!